5-8-2020

Intelligent Vehicular Perception of Non-Line-Of-Sight Environment using Visible Light Communication with Stereo Camera

Vignesh Varadarajan
Georgia State University

INTELLIGENT VEHICULAR PERCEPTION OF NON-LINE-OF-SIGHT ENVIRONMENT USING VISIBLE LIGHT COMMUNICATION WITH STEREO CAMERA

by

VIGNESH VARADARAJAN

Under the Direction of Ashwin Ashok, PhD

ABSTRACT

This thesis explores the use of stereo cameras to perceive immediate and non-line of sight roadway environments of a vehicle. The proposed system enables a "see-through-the-vehicle-in-front", functionality by combining scene perception with vehicle-vehicle communication. The fundamental idea of this work is to develop robust scene perception outcomes that can be communicated to other vehicles in vicinity, potentially using brake light to transmit and decode using cameras, conceptually similar to visible light communications. Through experimental evaluation of the prototype system, this work presents a proof-of-concept of non-line-of-sight (NLOS) perception in vehicles.

INDEX WORDS: Scene perception, Visible light communication, Non line of sight, See through, Camera receiver, LED transmitter

INTELLIGENT VEHICULAR PERCEPTION OF NON-LINE-OF-SIGHT ENVIRONMENT USING VISIBLE LIGHT COMMUNICATION WITH STEREO CAMERA

by

VIGNESH VARADARAJAN

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in the College of Arts and Sciences

Georgia State University

2020

INTELLIGENT VEHICULAR PERCEPTION OF NON-LINE-OF-SIGHT ENVIRONMENT USING VISIBLE LIGHT COMMUNICATION WITH STEREO CAMERA

by

VIGNESH VARADARAJAN

Committee Chair:	Ashwin Ashok
Committee:	Rajshekhar Sunderraman
	Shubham Jain

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

May 2020

DEDICATION

I dedicate my work to my parents who have constantly stood beside me with love and words of encouragement. I definitely would like to thank my friends who have wholeheartedly supported me no matter what.

ACKNOWLEDGMENTS

I am thankful to my advisor Dr. Ashwin Ashok for his guidance and insightful suggestions. I also thank my teammates Khadija Ashraf and Ryan Walden who have helped me greatly in completing my research. I would also like to thank Drs. Rajshekar Sunderraman and Shubham Jain for their valuable feedback of my research work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

VLC : Visible Light Communication

V2V : Vehicle to Vehicle

NLOS : Non Line-of-Sight

IoU : Intersection over Union

mAP : mean Average Precision

1 INTRODUCTION

1.1 Motivation

Ensuring the safety of drivers and passengers on the road has been a major concern that car manufacturers deal with. Of all the safety issues on the road distracted driving is one of the major causes of accidents in the United states [1]. 1 out of 5 cases of injuries is due to this. Another cause of concern is reckless driving causing 33% of all the fatal accidents. Both, distracted driving and reckless driving are examples where not only the participating vehicles get affected, but also the vehicles around them. These events could prove to be fatal for unassuming drivers on the road who are probably following all the rules. These unpredictable events are not realized quickly enough to get vehicles to safety. One of the main reasons for this is that the event is not in view. This is the problem that is being solved in this thesis.

The distracted drivers could put all the vehicles behind them at risk. Thus, we are introducing a "see-through" feature, essentially allowing all the vehicles on the road to see through the vehicles in the front. In this thesis, we use the term *non line-of-sight (NLOS) perception* to describe the the "see-through" feature.

1.2 Problem Statement

The problem we are trying to address here is that the vehicles on the road are prone to different safety issues due to the fact that they cannot see through the obstruction of their

view by other vehicles on the road.

This thesis posits that the see-through across vehicles can be established with timely detection of warnings and communicating them across vehicles. The hypothesis is that Non Line-of-Sight Perception (NLOS) is possible through intelligent inferencing from camera detection outputs and Vehicle to Vehicle (V2V) communication

1.3 Contributions

This thesis contributes the following research outcomes:

- Development of modules for NLOS perception in vehicles: Identified and developed modules required for detection of safety warnings in a scene.
- Development of brake light detection model: Trained a brake light detection model using a deep learning algorithm for the stereo camera.
- Design and implementation of visual perception for vehicular safety warnings. : Implemented modules that perceive the scene and generate safety warnings.

2 RELATED WORK

2.1 Vehicle Positioning

One of the main components involved in achieving NLOS or any V2V is locating the transmitting vehicles on the road. Now looking at the research in the past, the implementations mainly focus on various image processing techniques to detect the vehicles. Considering the "Looking at vehicles in the night: Detection and dynamics of rear lights" [2], it mainly focuses on applying image processing techniques to detect the bright tail lights in the night. Another work that focuses on localizing the LED source [3] uses LEDs on the road as static landmarks to further apply image processing techniques to relatively find the tailgates. To improve on this work, we make the location of an LED source independent of any additional infrastructure/equipment. We also use highly efficient and flexible deep learning algorithms to detect our LED source. Deep learning algorithms also ensure faster detection supporting our use-case.

2.2 V2V using Visible Light Communication

Vehicle to Vehicle communication using Visible light has been continuously evolving in recent times. Here, we see various works that use different transmitters and receivers to achieve this.

The research work in [4] uses LED panels as transmitters and image sensors for receivers. Another work [5], uses LIDARs and photo-diodes to establish the V2V channel. The research

[6] employs OCI sensors at the receiver end to pick up the visible light information. Here, the LED detection is again implemented using image processing techniques to detect the bright tail lights. The work [7] uses off the shelf LEDs and smartphone cameras to demonstrate V2V via VLC. While all these work establishes effective VLC channels, In the context of our work, NLOS, we need a more flexible implementation that would be able to detect the V2V Led sources on the go and receive information. This research proposes the use of components that already present in the vehicle. All vehicles are equipped with LED taillights that can act as a transmitter and almost all the vehicles these days are equipped with cameras, which we can use to our advantage as not only the receiver but also to perform scene analysis. The work [8] deals with the performance of VLC for different geometries and requires photodiode. We use object detection algorithms and camera receiver to detect any brake light and receive information. Further, these work [9], [10] and [11] provide detailed survey of work on Visible Light Communication.

2.3 Estimation of Depth

Estimation of the depth of various objects on the scene is very important to establish Non-Line of sight perception. This information is one of the main factors that help in understanding the actions of each object relative to other objects and thereby help in understanding the scene. In the work [12], LEDs are placed on the vehicles and it focuses on detecting this depth based on the intensity of light received at another following vehicle. The research [13] uses LED to photodiode communication to similarly assess the depth based on intensity. In

our, we are required to assess the depth of all the objects in the scene not only the LED data transmitting objects. Hence, we use off the shelf Zed stereo camera to get the point cloud information at all times for all objects in the scene.

2.4 Non Line-of-Sight Perception

NLOS essentially enables the vehicles to see/perceive information that is not in sight. We look at past research that focuses on certain modules that can be used to achieve NLOS. The work [14] compares the line of sight characteristics and security aspects of VLC to a practical V2V communication channel. One of the early works that discuss LOS and NLOS with respect to visible light communication is as part of the research [15]. Here NLOS mainly deals with capturing the LED information from a reflected surface rather than directly from the LEDs.

The work on "Augmented Vehicular Reality" [16] very closely deals with Non Line of Sight Perception as described in this research. This work calculates point cloud information in real-time and transmits this information across to other vehicles so that they further make their calculations to essentially understand the NLOS information. While this is a great implementation, it requires heavy computation and high-speed links to transfer this information. In our implementation, we use off the shelf Zed stereo camera to calculate the point cloud information and make inferences on the scene to generate warnings which are then transmitted to the other vehicles. We do not transmit the whole point cloud information thereby reducing the need extremely high data transfer rate.

3 SYSTEM DESIGN

To achieve NLOS, the fundamental idea of the proposed system design is to form a continuous loop of detecting and transmitting the warnings in all the vehicles. The system should be able to detect active and potential warnings in the scene and gauge its importance to act upon it. In our implementation, we divide these into sections as described in section 3.1. And in section 3.2 the implementation information of our system is detailed.

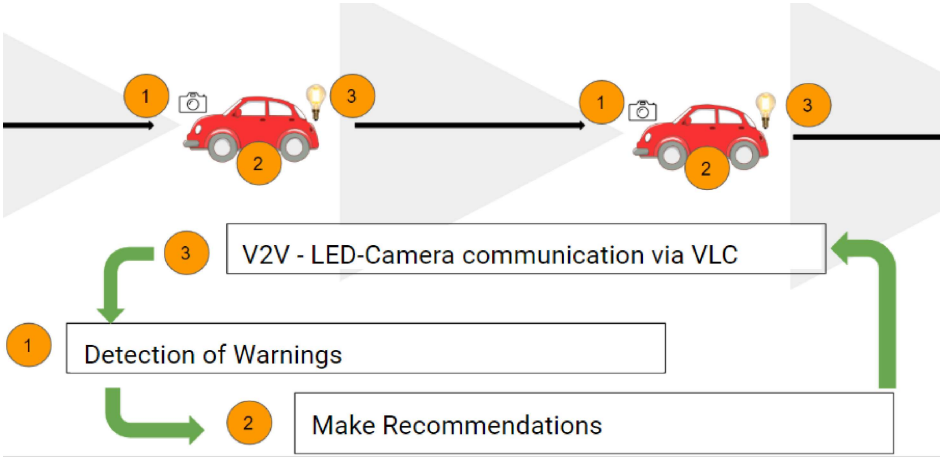

Figure 3.1 A conceptual diagram describing the proposed approach

3.1 Proposed Approach

The conceptual diagram describing our approach to address the NLOS problem is shown in Figure 3.1. It describes the continuous loop that enables vehicles to see through other vehicles on the road. As shown in the conceptual diagram, the system is mainly divided into three modules.

- Detection of warnings: This step aims to understand the scene in front of the car and

detect the relevant safety warnings

- Make recommendations: This step focuses on analyzing the received warnings, prioritizing them, and recommending the warning to the driver of the vehicle.

- V2V Communication: This step focuses on communicating the warning to the cars behind the current vehicle

These three sections are continuously in each and every vehicle to continuously look for safety warnings and communicating them effectively.

3.1.1 Detection of warnings

The first step in Non-Line of Sight Perception is understanding the scene and detecting the warning. As seen in Figure 3.2, the silver car tries to understand the scene. The silver car detects that the front car is lowering the speed. This detection can be categorized as a safety warning as the front car coming to halt could be an immediate safety issue for the silver car and any car that could be behind it. To have this system working, we would need a camera at the front of each car to capture the scene. The safety warnings are detected based on two main components:

- Object Detection on the scene

- Further analysis of the type/location/distance of the detected objects

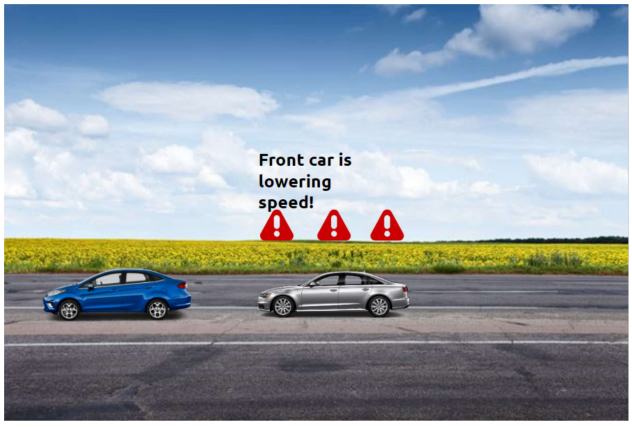

Figure 3.2 Figure depicting the Detection of warnings section

Figure 3.3 Figure depicting the Make recommendations section

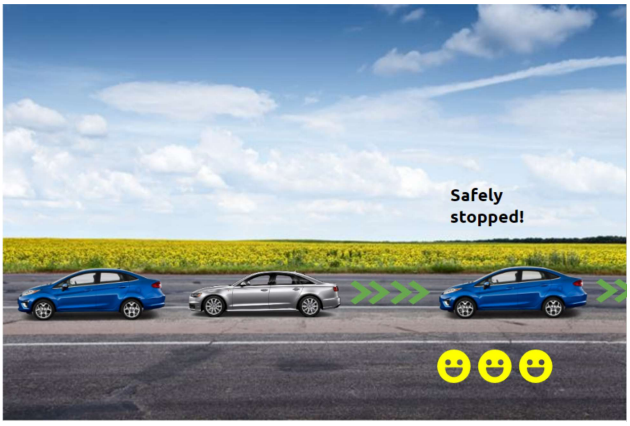

Figure 3.4 Figure depicting the V2V Communication section

3.1.2 Make Recommendations

The second step in Non-Line of Sight Perception is deciding which of the detected warnings are important to make a recommendation as well as communicate it to the other vehicles. As seen the Figure 3.3, the silver car detected that the front car had lowered the speed and it has recommended that the car comes to a halt. The silver car halts in time. This step includes four functions:

- Analysis of all warnings: Decide which of the detected warnings require immediate attention
- Prioritize warning: Of the warnings that require immediate attention, we prioritize warnings based on their importance.
- Pick the highest priority warning: Picks the most important warning that needs to be acted upon.
- Make a recommendation: This step makes the recommendation of the warning to the driver and readies the warning for transmission

3.1.3 V2V Communication

This step virtually enables the see-through feature by communicating the warning from the previous step across different vehicles. Since our system uses LED-Camera communication via Visible Light communication to transmit the information what we need are 2 LEDs at the back of the car for transmission. We reuse the camera in the front to receive the VLC

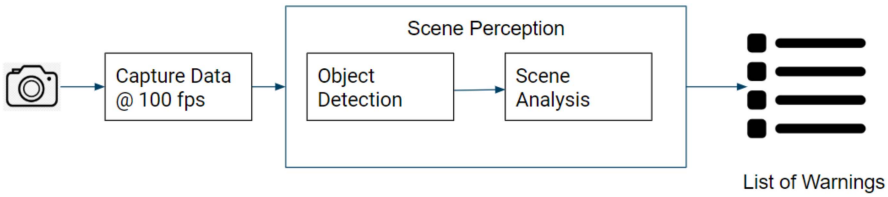

Figure 3.5 System Architecture Diagram that describes the flow of components implemented in this research

Data. Figure 3.4 shows that the silver car transmits the warning across and a blue car entering the scene could safely stop in time.

3.2 System Components

The notion of NLOS perception can be achieved only through a marriage between scene perception analysis and V2V communication. The fundamental notion of the research space addressed in this thesis is to use VLC for communicating between the vehicles, using the brake light as transmitters. However, we note that this thesis only focuses on developing the scene perception analysis. While the key motivation for perceiving and detecting the brake light, and its light emission status, are the motivation for visible light communication between the vehicles, this thesis does not focus on building the VLC components in particular.

The key components involved in this work starting from capturing the scene to identifying the warnings has been described in Figure 3.5 and the following sections describe each of the steps in detail.

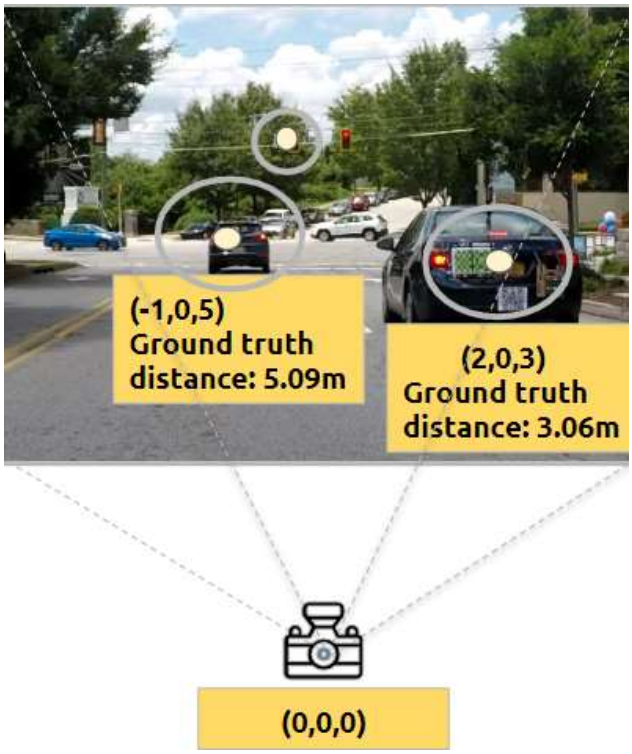

Figure 3.6 Annotated illustration of detected objects using ZED camera and distance estimates using stereo-vision. We also have noted the ground truth for each vehicle measured physically during data collection.

3.2.1 Data Collection

Data collection steps essentially capture all the data that is required for the rest of the processing pipeline. In our case, for the data collection, we use Zed cameras. A zed camera is an off the shelf stereo camera. A stereo camera is built with two cameras, with a left and a right view. It uses these two views to derive the real-world point cloud information. The Zed camera in our implementation is connected to an NVIDIA Xavier device. This device acts as the controller that sends the signal to capture the data.

As shown in Figure 3.6, the zed camera provides point cloud information for all the pixels in the image. We get the real-world coordinates relative to the camera. The camera is

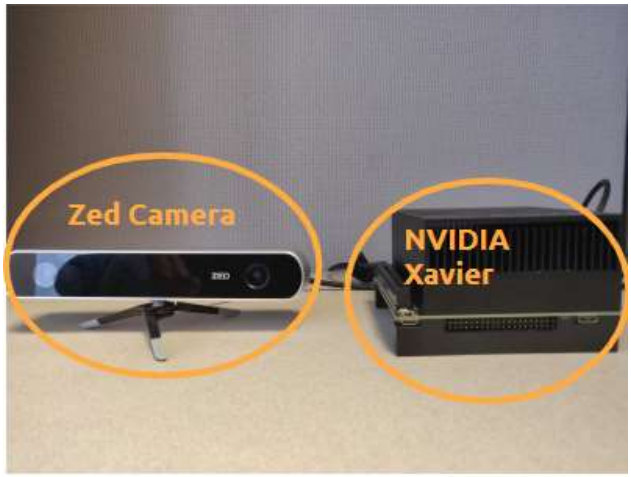

Figure 3.7 Zed Camera connected to Jetson Xavier

assumed to be at (0,0,0). Now, for every pixel on the image, the zed comes with an in-built API that can give the real-world (x, y, z) values. From this information, we calculate the Euclidean distance to get the distance between the camera and any object in the scene. Zed camera also provides a 100 fps capture rate @ VGA resolution which is 600x480 pixels. [17] As the first step in this implementation, we let the camera run for 1 second and it captures 100 frames in that time span along with its point cloud information. We use this information for further processing in the next steps.

3.2.2 Scene Perception - Object Detection

This is the first part of Scene Perception. To be able to perceive the scene, detecting the objects on the scene plays a very important role. To achieve this, we use deep learning algorithms. They are advantageous due to their flexibility in customizing the type of objects that they would detect and gives great processing times. The table 3.1 shows the processing times provided by different Deep learning algorithms.[18] [19] Each of these time information

Algorithm	FPS
Mobile Net	3.57
Faster RCNN	1.2
YOLO	7.5
Tiny YOLO	15

Table 3.1 Processing times for different algorithms

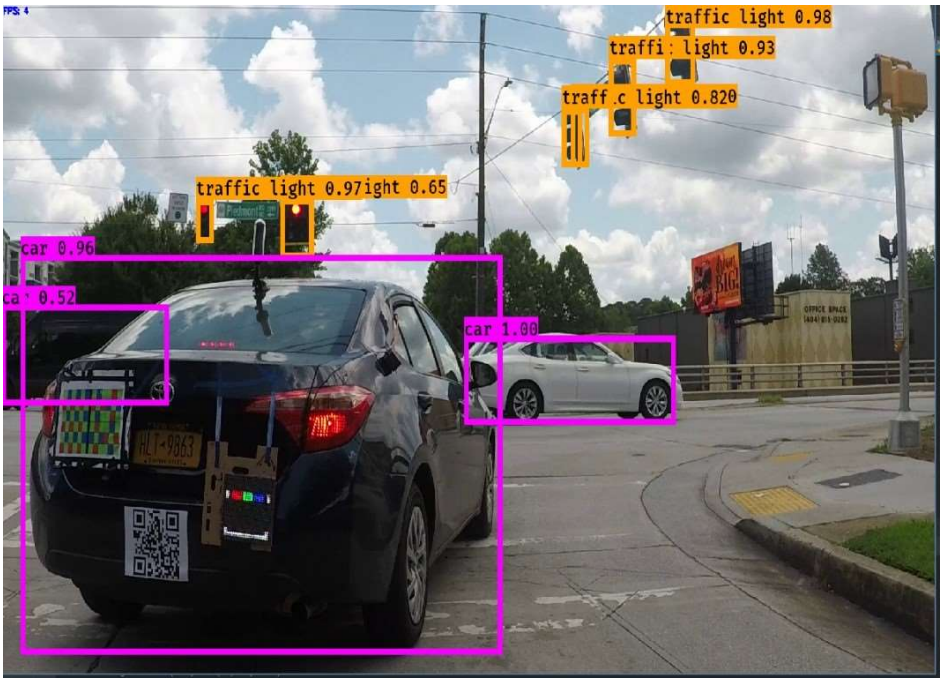

Figure 3.8 Figure shows the output of Tiny - YOLO

has been obtained by running these algorithms on NVIDIA Jetson Xavier and NVIDIA GeForce GTX 1060. While Mobile Net, Faster RCNN comes with great accuracy. YOLO [20] trades it off with great processing time. With good training, it was seen that YOLO could provide both good accuracies along with the good processing time, Tiny YOLO is a lighter version of YOLO, it provides faster implantation rates.

Both YOLO and tiny YOLO come with a pre-trained model that efficiently detects up to 90 objects. Since our implementation requires the objects on the road to be detected, the following objects such as traffic signal, people, vehicles, stop signs are helpful.

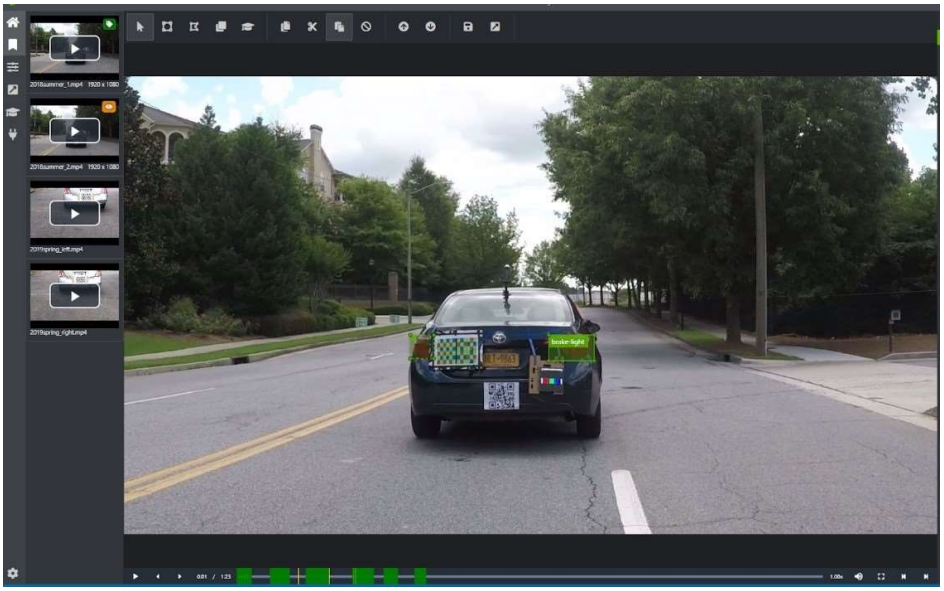

Figure 3.9 Tagging and Labelling done on Microsoft VoTT

Figure 3.8 shows the output of the Tiny YOLO Object detection algorithm run on an image. One of the main objects that we are missing from this model is the Brake-lights. The need for Brake-lights in our system is two-fold. We intend to use brake-light not only to detect its status to derive any impending safety warning since we are also using the same LEDs as the transmitter in LED-Camera communication via Visible Light communication. Thus, to be able to detect the brake-lights in real-time, we custom-trained the YOLO model. To do this, we successfully tagged and labeled over 3245 brake-lights. Microsoft Visual Object Tagging Tool – Microsoft (VoTT) (shown in image 3.9) is a tool that we used to tag the brake lights.

We used Keras, a python framework on TensorFlow to train and execute our model. [21] The dataset that we have created is an evolving dataset that we intend to make it available to the public. The image 3.10 shows the output of the object detection algorithm after

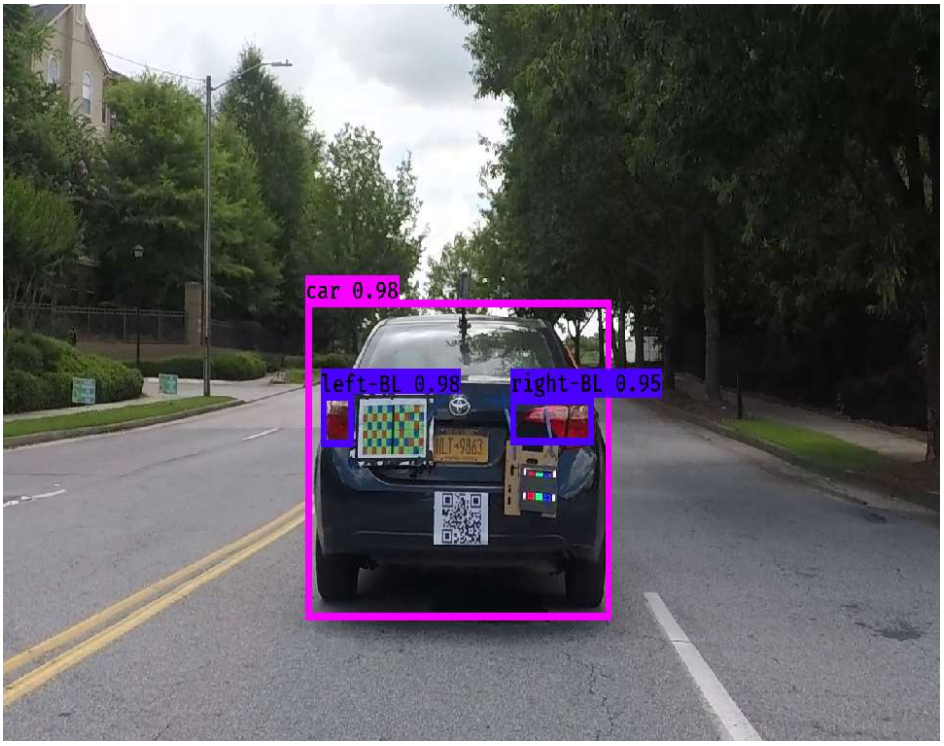

Figure 3.10 Output of YOLO after training it for brake-lights

training for the brake lights. It can also be seen that, with respect to the coordinates of the car that the brake lights belong to, we categorize the detected brake lights to be either left or the right one. As the data is being captured at 100fps, we do not see a massive difference between the location of objects on the scene between each of these frames. Based on trial and error, it was found that it is enough to run these algorithms on three of the hundred frames. We use the 1st, 31st, and the 61st frame.

3.2.3 Scene Perception : Scene Analysis

Scene Analysis is the final step of Scene Perception that follows object detection. The main functions of this stage include analysis of the objects detected on the scene and its relative

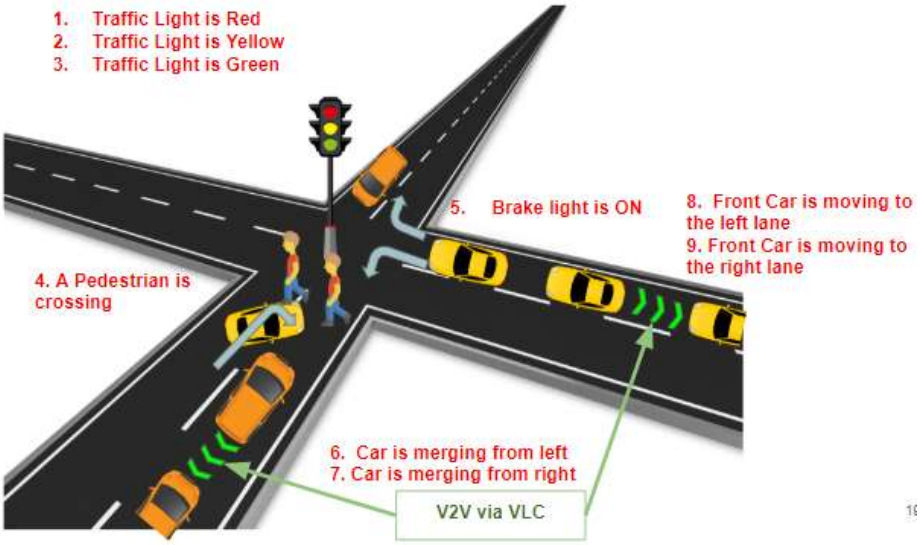

Figure 3.11 List of all the warnings

distance from the camera and deriving the appropriate warnings. The image 3.11 shows the warnings that have picked to demonstrate our system.

At this stage in our system, we have objects detected and localized in the scene. The following sections describe in detail how each of the above warnings is derived. The algorithm 1 describes the scene perception flow.

Algorithm 1 PercieveScene(frames)

```
Input : 100 frames captured by the camera
Output : JSON with attributes
scenceData := Scene(cars, trafficLights, trafficSigns, pedestrians);
scenceData := runYOLOScenePecception(frames)
brakeLights := runYOLOBrakeLightDetection(frames)
packetBits := receiveVLCData(frames, brakeLights)
analyzeCarsAndBrakeLights(scenceData.cars, brakeLights)
analyzeTrafficLights(scenceData.trafficLights)
analyzeTrafficSigns(scenceData.trafficSigns)
analyzePedestrian(scenceData.pedestrians)
analyzeLaneInformation(scenceData, brakeLights)
return JSON(scenceData, brakeLights, packetBits)
```

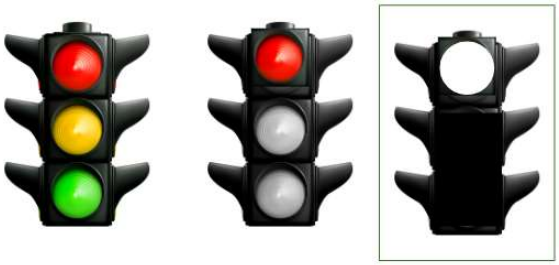

Figure 3.12 Traffic Light Analysis

3.2.3.1 Traffic Light Analysis

Once the traffic lights are detected on the scene, the next step would be to check if the traffic light is red/green/yellow. To achieve this, we apply a mask on the image. The mask is a color range that is used to filter the image. If there are any pixels in the image within the selected range, then those pixels would be returned as 1 after applying the mask else 0. For example, the mask range used for the color red is Upper Mask: (180,255,255) and Lower Mask: (175.50.20). In the Figure 3.11, the first image shows the traffic light object. The second image shows the part of the image that falls in the range (in red). The third part shows the final output where only the red part of the object is white in color, everything else is zero. Now, with this 2D matrix, we calculate the number of 1s. If this number of 1's crosses a certain threshold, then we confirm that that is the color of the traffic light. The threshold we use here is 1%. If the number of 1s cross 1% of all the output values, then we confirm that the color exists. The output of this step would be a warning generated such as, "Traffic Light is Green", distance = 5 meters.

This code snippet shows the steps as described in Figure 3.13.

```
def detect_red(img, desired_dim, Threshold=0.01):

    # lower mask (0-10)
    lower_red = np.array([8,70,50])
    upper_red = np.array([10,255,255])
    mask0 = cv2.inRange(img, lower_red, upper_red)

    # upper mask (170-180)
    lower_red = np.array([175,70,50])
    upper_red = np.array([180,255,255])
    mask1 = cv2.inRange(img, lower_red, upper_red)

    # red pixels' mask
    mask = mask0+mask1

    # Compare the percentage of red values
    rate = np.count_nonzero(mask) / (desired_dim[0] * desired_dim[1])
    if rate > Threshold:
        return True
    else:
        return False
```

Figure 3.13 Traffic signal code snippet

3.2.3.2 Lane Analysis

To detect many of the warnings listed in the previous section, it is important that we detect the lanes that each of the objects is present in. They greatly help us in deciding what warnings require immediate action. In our system, we detect the lanes based on the geometrical location of the object relative to the car. Assuming that the camera has the view of the lane that it belongs to as shown in Figure 3.14, we geometrically create a triangle to categorize the objects into the lanes they belong to. This triangle is drawn by connecting the points (0, width/2), (1/3 width, height), and (2/3 width, height). Now, if the object falls within the triangle, we conclude that it belongs to lane 0 (same lane). If it falls to the

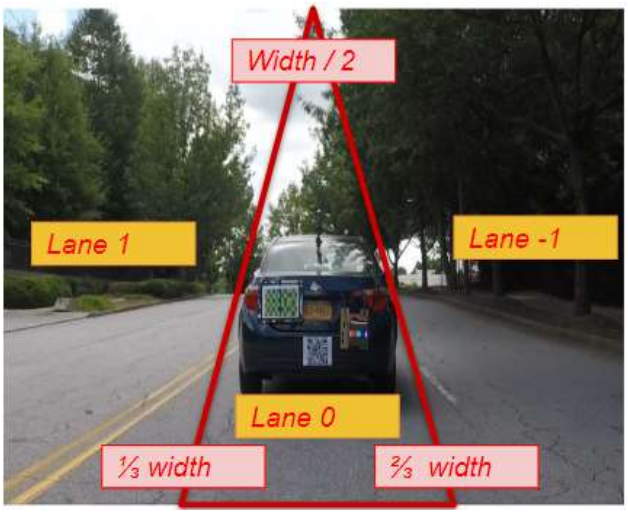

Figure 3.14 Lane Analysis : Figure describing the geometry used to categorize objects into their respective lanes

left of the triangle, then we conclude that it belongs to lane 1 (left lane) else if right then lane -1 (right lane). These calculated simply by checking if the point falls above or below the lines of the triangle. A sample code snippet is shown in Figure 3.15.

Now once the lane information is gathered, by comparing this information among multiple frames we would be able to detect the following warnings,

- A car is entering the lane from the left
- A car is entering the lane from the right
- The front car is moving to the left lane
- The front car is moving to the right lane

```
def get_lane_for_coordinates(x, y):
    # Since it is VGA
    h = 480
    w = 640
    p = np.array([x, y])
    is_above = lambda p, a, b: np.cross(p - a, b - a) < 0
    a_left = np.array([w / 3, h])
    b_left = np.array([w / 2, 0])
    a_right = np.array([2 * w / 3, h])
    b_right = np.array([w / 2, 0])

    above_left = is_above(p, a_left, b_left)
    above_right = is_above(p, a_right, b_right)
    lane = None

    if above_left and above_right:
        lane = 0
    elif not above_left:
        lane = -1
    elif not above_right:
        lane = 1

    return lane
```

Figure 3.15 Lane Analysis code snippet

3.2.3.3 Other warnings and the output

Similar to the detection of traffic light warnings, we would be able to get the brake light information as well. We apply a similar mask for red to get the brake light status. Since we gather the lane information in the previous step, we would also be able to get the pedestrian and road sign warnings. We would be able to tell how far a pedestrian is and also tell the lane they belong to. As of now, our system detects that road signs exist in the scene, but more work is needed to categorize the road signs. At the end of this step, the algorithm returns a JSON that encompasses all this information. A sample format of the JSON is

```
{
  "cars": [
    {
      "id": 1,
      "is_any_car": false,
      "relative_speed": 5,
      "distance_closer_to": 5,
      "depth_closer_to": 7,
      "is_brake_light_on": true,
      "is_hazard_light_on": false,
      "lane_closer_to": 0,
      "is_leaving_lane_left": false,
      "is_leaving_lane_right": false,
      "is_merging_lane_from_right": false,
      "is_merging_lane_from_left": false
    }
  ],
  "traffic_sign": {
    "is_any_sign": true,
    "is_red": true,
    "is_yellow": false,
    "is_green": false,
    "distance_closer_to": 19,
    "depth_closer_to": 20,
    "speed": 45
  },
  "pedestrians": [
    {
      "id": 1,
      "distance_closer_to": 27.5,
      "depth_closer_to": 20,
      "lane_closer_to": 0
    },
    {
      "id": 2,
      "distance_closer_to": 27.5,
      "depth_closer_to": 20,
      "lane_closer_to": 0
    }
  ],
  "packets_from_cars": []
}
```


Figure 3.16 Sample message in JSON format

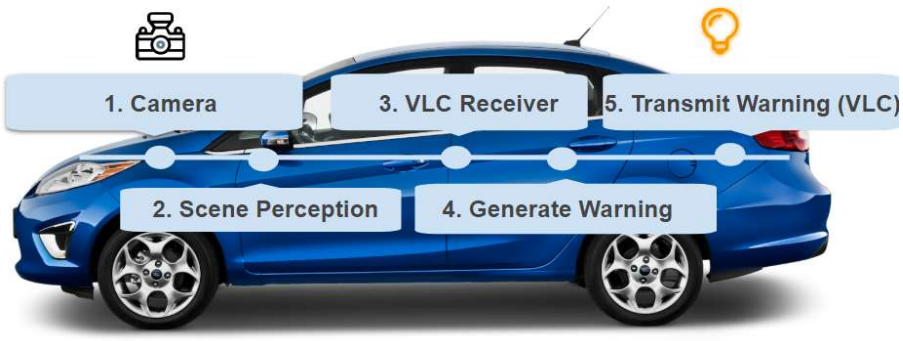

Figure 3.17 Describes the steps involved in NLOS

shown in Figure 3.16.

3.2.4 Prototype Implementation

While this thesis focuses on detection of warnings, the entire prototype does include the V2V communication component. This implementation as seen in the Figure 3.17 describes the 5 main sections that form the prototype.

- Camera – Capturing the data
- Scene Perception – Detection of safety warnings
- VLC Receiver – Receive VLC Data
- Generate Warning – Generate a warning that needs to be recommended/transmitted
- Transmit Warning (VLC) – Using the LEDs at the back of the car, transmit the warning.

4 EVALUATION

In this section, we have performed a detailed analysis of our implementation. We evaluate the scene perception module that has been prepared for NLOS. The results of this section determine the effectiveness of our system in determining the safety warnings on the road.

4.1 Experimental Setup

The main components need for the setup is the Zed camera, the LED lamps, NVIDIA Xavier, and the Vehicles.

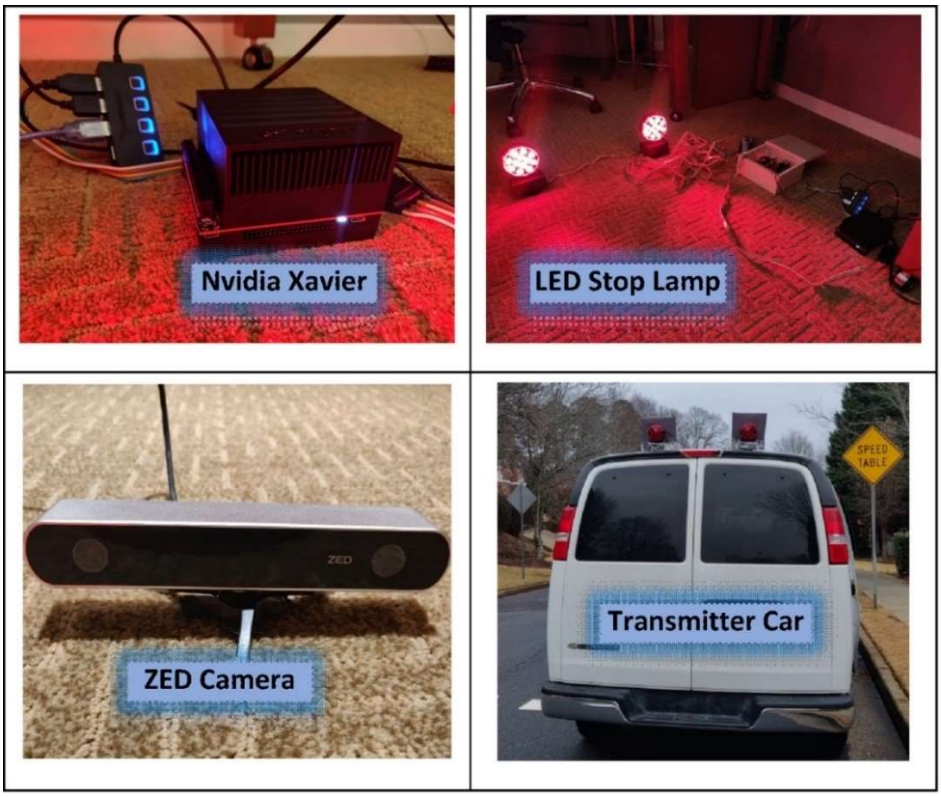

Figure 4.1 Main components in the setup

As described the Figure 4.2, the Camera is placed at the front of the car. The Scene

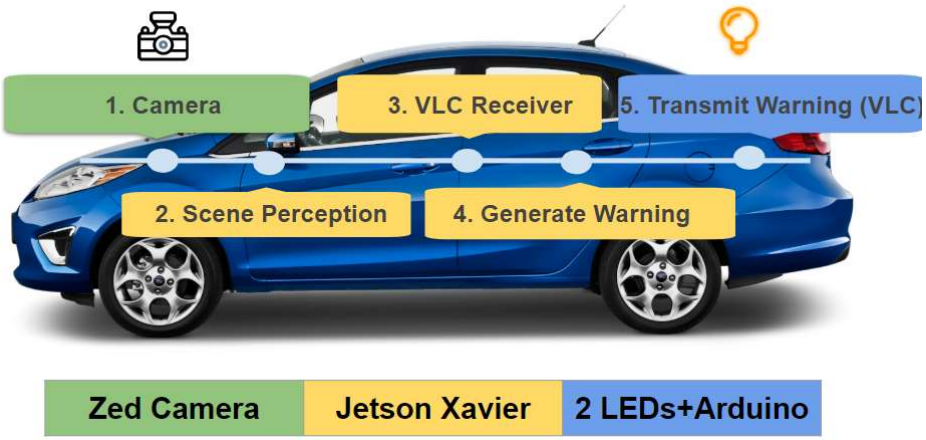

Figure 4.2 Figure shows the components used for each of the steps

Perception, VLC Receiver and Generate warnings step require a processor equipped with GPU for fast computations. For this purpose, we use NVIDIA Jetson Xavier. And finally, for LED to camera communication, we fit two LEDs at the top of the back of the car. This coupled with Arduino to control it, form the LED transmitter. This entire setup is replicated in two vehicles (a leading car and a following to car) to demonstrate/evaluate the Non-Line of Sight feature.

4.2 Dataset for Evaluation

- 20 Videoes approximately 2 hours of on-road data collected
- Public dataset with 40 videos [22]

4.3 Object Detection

We describe the accuracy of our Brake-light detection model using the mean Average Precision(mAP) metric.[23] The AP value is calculated by computing the Area Under Curve

Figure 4.3 Sample of Data Collected

(AUC) in the precision vs recall graph.

We define Precision and Recall as follows:

$$Precision = TP/(TP + FP) \tag{4.1}$$

$$Recall = TP/(TP + FN) \tag{4.2}$$

where TP = True Positive, FP = False Positive and FN = False Negative. This AP is calculated for multiple IoUs. IoU, the Intersection over Union metric, gives the percentage overlap between the predicted object area and the ground truth. This has been computed on 170 images with 415 objects in the test dataset. A common method to arrive at mAP, as shown in table 4.1 is by calculating the mean of all APs for IoUs between 0.5 and 0.95

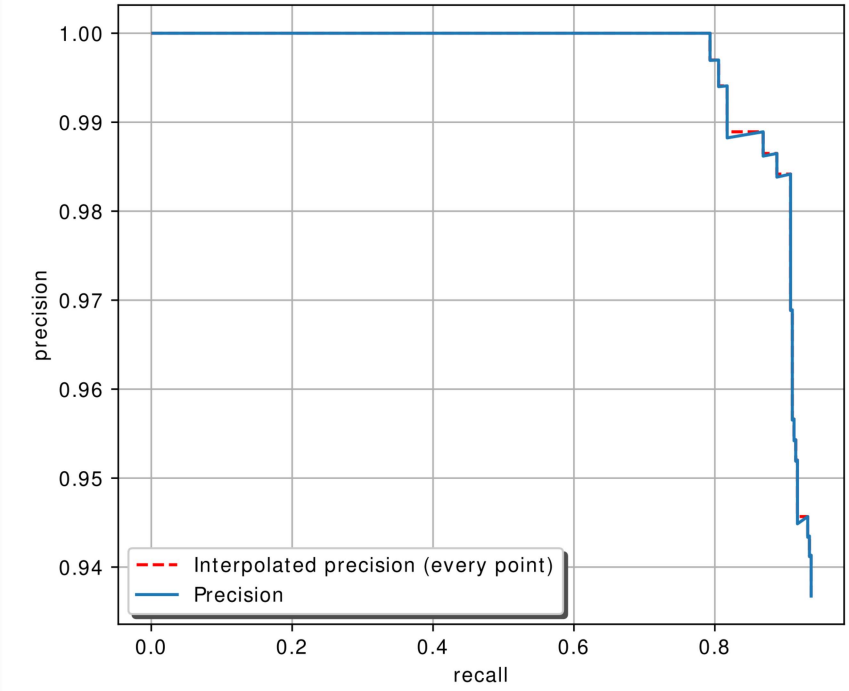

Figure 4.4 Brake Light Detection Training : Precision vs Recall graph IoU = 0.50, AP = 93.40

with a step of 0.5. Figures 4.4, 4.5 and 4.6 shows the graphs for IoUs 0.50, 0.75 and 0.95 respectively.

IoU (%)	50	55	60	65	70
AP	93.40	93.40	93.29	93.29	92.76
IoU (%)	75	80	85	90	95
AP	92.22	91.68	90.62	89.49	72.40
mAP = 90.255					

Table 4.1 Average Precision for different IoUs

4.4 Scene Perception

To evaluate the scene perception module, the warnings detected in real-time were compared with an expected warning. Based on the real-time data collected, expected warnings

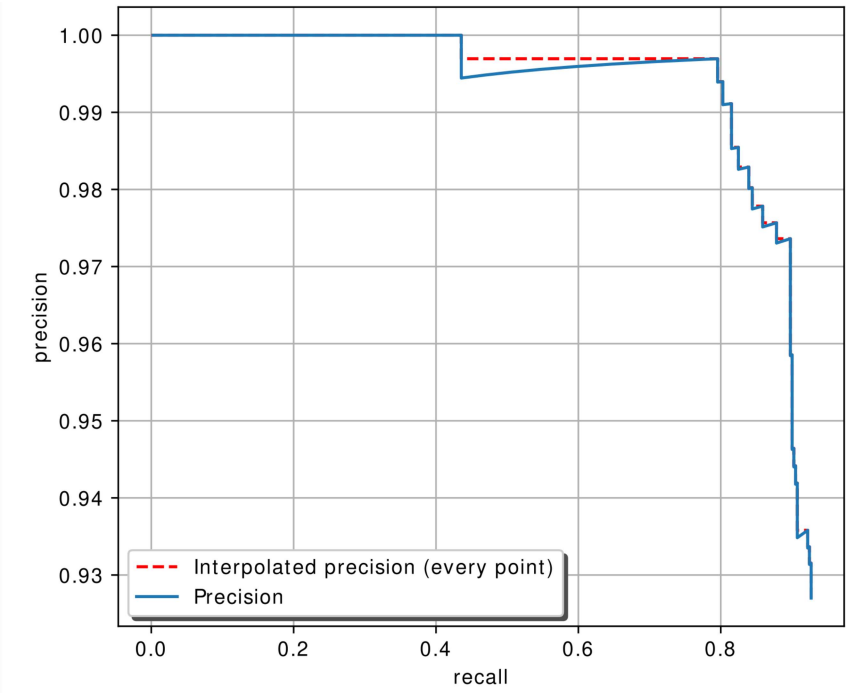

Figure 4.5 Brake Light Detection Training : Precision vs Recall graph IoU = 0.75, AP = 92.22

that form the ground truth data were carefully generated by manually going through the stereo captured videos. The warnings that were expected from these videos were categorized separately for a better understanding of the performance and further analysis. Also, it is important to note that the number of warnings in each category is not distributed uniformly as they have been derived from the dataset collected in real-time. Figure 4.7 presents the efficiency of the scene perception algorithm. We have picked the most commonly occurring safety warnings.

We denote our results to show the following information for each category of warning: Correct Detection, Wrong Detection, and Not Detected. As per our analysis, the majority of the cases where the warnings were not detected are directly linked to the detection of

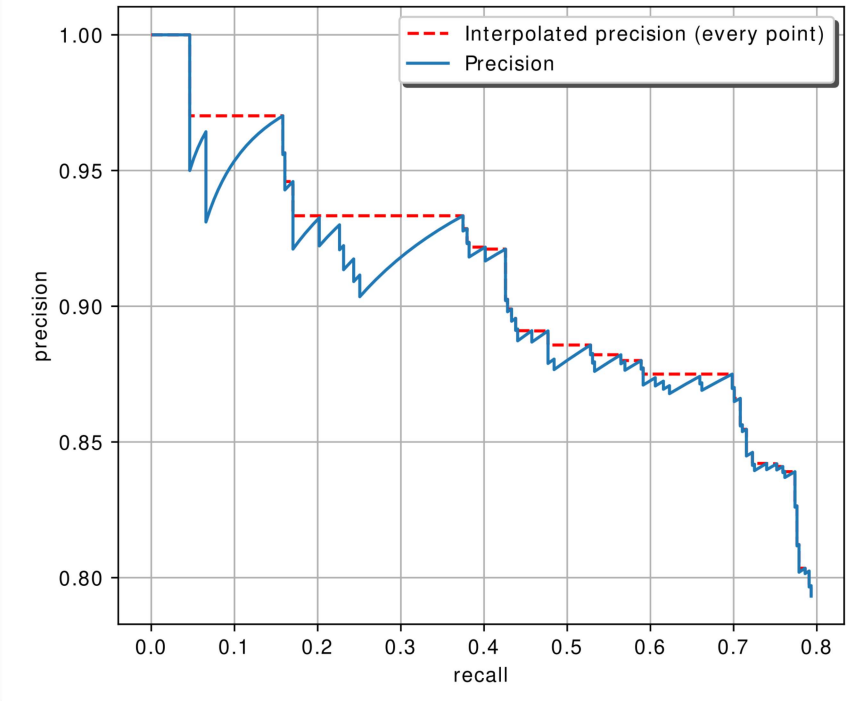

Figure 4.6 Brake Light Detection Training : Precision vs Recall graph IoU = 0.95, AP = 72.40

objects on the scene. If the object is not detected, the perception takes a hit. The accuracy of the object detection algorithm is detailed in the next section. In cases where the warning has been wrongly detected, we notice that this is mainly due to segmentation errors. The overall accuracy of scene perception achieved is 93.02%

4.5 Timing Information

The timing information provided in Table 4.2 is a result of averaging time data received from over 2500 end to end runs. The timing information for each module was calculated by calling the functionality that provides current time which is part of the "time" library [24] that is included in python3. This function is called at the beginning and end of each step and the

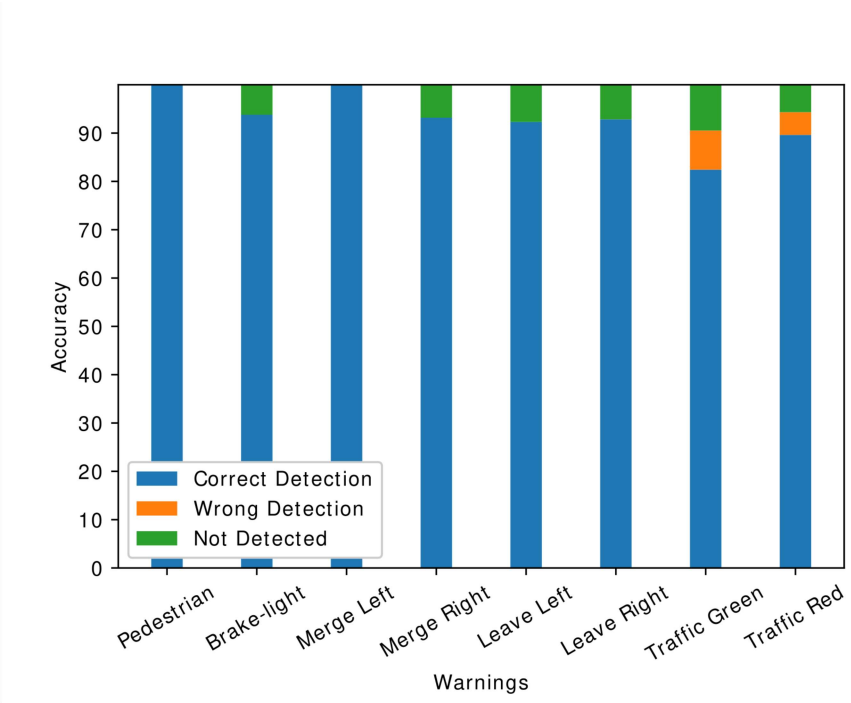

Figure 4.7 Scene Perception Accuracy

Module	Details	Average (seconds)
Camera Capture	Capture 100 frames @ 100 fps	1
Scene Perception	a) Object Detection - Tiny YOLO (0.04 * 3 = 0.12)	0.16
	b) Object Detection - YOLO (0.08 * 3 = 0.24)	0.298
	c) Traffic Light / Brake Light etc..	0.012
	d) Gather Lane Information	0.0001
Total Scene Perception Analysis time with Image Capture		1.4701
Total Scene Perception Analysis time minus Image Capture time		0.4701

Table 4.2 Timing Information

time difference is calculated accordingly to get the time taken for each step. Each module has been identified as discussed in the section - System Architecture. The object detection

and scene perception steps are all performed on three frames. VLC Demodulation is run on all 100 frames that are captured. It can be observed that the majority of the processing time is taken by the two object detection steps. This can be further reduced if we switch to a device with a faster GPU.

5 CONCLUSIONS

This thesis discusses the design and successful development of modules required for Non Line of sight perception. It presents a brake light detection model that has been custom trained for this use-case and is operable in real-time with an accuracy greater than 90%. This work successfully implements visual perception for vehicular safety warnings, also with accuracy greater than 90%. Also, this work was successfully demonstrated at the IEEE Vehicular Networking Conference 2019 [25].

As part of future work, the existing implementation can be extended to detect many other safety warnings such as emergency vehicles, actual and potential collisions, a variety of road signs, etc.. Another improvement could be improving the time taken for by the deep learning object detection algorithms. On Pascal Titan X, YOLO is capable of processing up to 45 fps and Tiny YOLO up to 220 fps. With these improved times, the generation of safety warnings could happen at a much faster rate.

REFERENCES

[1] Car accident statistics in the u.s. https://www.driverknowledge.com/car-accident-statistics/, 2020.

[2] Ravi Kumar Satzoda and Mohan Manubhai Trivedi. Looking at vehicles in the night: Detection and dynamics of rear lights. *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[3] Byung Wook Kim and Sung-Yoon Jung. Vehicle positioning scheme using v2v and v2i visible light communications. In *Vehicular Technology Conference (VTC Spring), 2016 IEEE 83rd*, pages 1–5. IEEE, 2016.

[4] Trong-Hop Do and Myungsik Yoo. A multi-feature led bit detection algorithm in vehicular optical camera communication. *IEEE Access*, 7:95797–95811, 2019.

[5] Bastien BÃ©chadergue, Wen-Hsuan Shen, and Hsin-Mu Tsai. Comparison of ofdm and ook modulations for vehicle-to-vehicle visible light communication in real-world driving scenarios. *Ad Hoc Networks*, 94:101944, 2019.

[6] Yuki Goto, Isamu Takai, Takaya Yamazato, Hiraku Okada, Toshiaki Fujii, Shoji Kawahito, Shintaro Arai, Tomohiro Yendo, and Koji Kamakura. A new automotive vlc system using optical communication image sensor. *IEEE photonics journal*, 8(3):1–17, 2016.

[7] P. Ji, H. Tsai, C. Wang, and F. Liu. Vehicular visible light communications with led taillight and rolling shutter camera. In *2014 IEEE 79th Vehicular Technology Conference*

(VTC Spring), pages 1–6, May 2014.

[8] P. Luo, Z. Ghassemlooy, H. Le Minh, E. Bentley, A. Burton, and X. Tang. Fundamental analysis of a car to car visible light communication system. In *International Symposium on Communication Systems, Networks Digital Signal Processing (CSNDSP)*, pages 1011–1016, July 2014.

[9] A. M. Cailean, B. Cagneau, L. Chassagne, V. Popa, and M. Dimian. A survey on the usage of DSRC and VLC in communication-based vehicle safety applications. In *IEEE Symposium on Communications and Vehicular Technology in the Benelux (SCVT)*, pages 69–74, November 2014.

[10] Trong-Hop Do and Myungsik Yoo. An in-depth survey of visible light communication based positioning systems. *Sensors*, 16(5):678, 2016.

[11] Nam Tuan Le, Mohammad Arif Hossain, and Yeong Min Jang. A survey of design and implementation for optical camera communication. *Signal Processing: Image Communication*, 53:95–109, 2017.

[12] Takaya Yamazato and Shinichiro Haruyama. Image sensor based visible light communication and its application to pose, position, and range estimations. *IEICE transactions on communications*, 97(9):1759–1765, 2014.

[13] V. T. B. Tram and M. Yoo. Vehicle-to-vehicle distance estimation using a low-resolution camera based on visible light communications. *IEEE Access*, 6:4521–4527, 2018.

[14] Bugra Turan and Seyhan Ucar. Vehicular visible light communications. *Visible Light Communications*, page 133, 2017.

[15] Pengfei Luo, Zabih Ghassemlooy, Hoa Le Minh, Edward Bentley, Andrew Burton, and Xuan Tang. Performance analysis of a car-to-car visible light communication system. *Applied Optics*, 54(7):1696–1706, March 2015.

[16] Hang Qiu, Fawad Ahmad, Fan Bai, Marco Gruteser, and Ramesh Govindan. Avr: Augmented vehicular reality. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '18, page 81–95, New York, NY, USA, 2018. Association for Computing Machinery.

[17] Zed stereo camera. `https://www.stereolabs.com/zed/`, 2015.

[18] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[19] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[20] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[21] qqwweee. keras-yolo3. `https://github.com/qqwweee/keras-yolo3`, 2018.

[22] Zhiyong Cui, Shao-Wen Yang, and Hsin-Mu Tsai. A vision-based hierarchical framework for autonomous front-vehicle taillights detection and signal recognition. In *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, pages 931–937. IEEE, 2015.

[23] Sergio Lima Netto Rafael Padilla and Eduardo A. B. da Silva. Survey on performance metrics for object-detection algorithms. 2020.

[24] Python Software Foundation. time : Time access and conversions. `https://docs.python.org/3/library/time.html`, 2020.

[25] V. Varadarajan, K. Ashraf, and A. Ashok. Demo : Intelligent vehicular perception of non-line-of-sight environment using visible light communication with stereo cameras. In *2019 IEEE Vehicular Networking Conference (VNC)*, pages 1–2, 2019.